#4

EL684297009US

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

U.S. PATENT APPLICATION

for

METHOD AND SYSTEM FOR LINK FABRIC ERROR DETECTION AND MESSAGE FLOW CONTROL

Inventors

Robert C. Frisch, a citizen of the United States
residing at 29 Pine Ridge Road,
Westford, Massachusetts 01886

Bryan D. Marietta, a citizen of the United States
residing at 13117 Boomer Lane,
Austin, Texas 78729

Daniel L. Bouvier, a citizen of the United States
residing at 8112 Asherton Cove,
Austin, Texas 78750

METHOD AND SYSTEM FOR LINK FABRIC ERROR DETECTION AND MESSAGE FLOW CONTROL

REFERENCE TO RELATED APPLICATIONS

This application claims the benefit of priority of United States Provisional Patent Application Serial No. 60/175,856, filed January 13, 2000, entitled "Rio Bus Protocol," the entirety of the teachings of which are incorporated by reference.

BACKGROUND OF THE INVENTION

The current development of computer and embedded systems is burdened by divergent requirements. On one hand performance is expected to increase exponentially, while on the other hand the price of components is driven by market factors to continuously decrease. Developers are burdened with the challenge of increasing device density, or functional density, while decreasing the board space and ultimately the floor space that the components require. The physics of signal handling and transmission, and the need for standardized interconnects and scalable systems, impose further bounds on system development.

Many existing computer systems are built around a single high-speed common bus. When fast digital devices are connected to the bus, they can communicate more quickly, e.g., with higher frequency bus signals. However, as device bandwidth demands increase, each consumes a greater portion of the total bus bandwidth, so that the addition of a small number of high-speed devices quickly offsets overall bus bandwidth gains. Furthermore, interfacing such high bandwidth devices poses additional hardware problems in the physical environment of a typical computer system.

Typically, the connection between microprocessors and peripherals in a computer system has through a hierarchy of shared buses. Devices are placed in an appropriate position in the hierarchy, given their respective levels of performance. Low performance devices are placed on lower performance buses, that are bridged to the higher performance buses, so as to not burden the higher performance devices. Bridging is also used to interface legacy interfaces.

Objectively, the need for higher levels of bus or interconnect performance is driven by two key demands. First is the need for raw data bandwidth to support higher performance peripheral devices. Second is the need for higher degrees of system concurrency. Increasingly, system designers are relying on distributed direct memory access (DMA) and distributed processing to meet these demands.

Over the past several years, the shared multi-drop bus has been exploited to its full potential. Many techniques have been applied to enhance the effective bus bandwidth, such as increasing frequency, widening the interface, transaction pipelining, split transaction, and out of order completion. Continuing to work with a bus in this manner introduces several design issues. Increasing bus width, for example, raises conflicts with physical limitations on the maximum achievable frequency, in part, because of the difficulty of maintaining skew tolerance between signals. More signals also requires more complex hardware, e.g. pins and interface logic, resulting in higher product costs and fewer supported interfaces per device.

Another way to circumvent the bandwidth limitations of a multi-computer system having a common bus is through distributed communications. In this approach, the components of a system are interconnected by multiple local buses. Both the nature and number of local buses can be varied to match the communications needs of a particular system.

Yet another approach is a fabric-based interconnect system. Here, buses are not used as communications paths in the traditional sense. Rather, information is routed between end nodes via fabric of communications links and other nodes. Information traffic in fabric-based systems typically consists of messages having a distinct format and following a protocol. The message format may include routing or other information, in addition to data or "payload," such that each packet can be routed at high speed along the links to its destination. As applied to DMA systems, protocols must permit messages to be driven by the source (which requires access not only to a target, but also visibility into the target's address space), whereas other messages for other transactions or operations are typically steered by the destination.

In order to properly manage the transmission of message packets between hops of a fabric-based system, it is necessary to exchange control signals in a form of "handshake" signaling. Some prior art systems utilize additional, side band signal pins to support the handshake. Other systems have protocols requiring the explicit exchange of control message packets between nodes.

Fabric-based systems offer a number of distinct advantages, since the interconnect fabric may be set up or dynamically reconfigured to route data along available lines, and to provide alternate pathways between endpoints, so that messages, control operations and data may be routed on many paths without collision. Thus the distributed routing of a fabric interconnect message-passing system is advantageously applied to applications, such as distributed direct memory access (DMA) or distributed processing, to further enhance the efficiencies of those approaches.

However, a message-passing system requires adherence to strict protocols for its operation, and is still subject to bandwidth limitations and varying traffic along its constituent links. Moreover, the passing of messages along different links raises many potential failure points, where message corruption or deadlock may occur. Further, the prior art use of side band signal pins to support handshake signaling comes at a cost: more pins, more logic and more board "real estate." The alternative use of explicit control message packets for this purpose can prove expensive from a bandwidth and latency point of view.

An object of this invention is to provide improved digital data systems and methods of operation thereof.

A further object is to provide such systems and methods as provide improved control of traffic in a message-passing digital data system.

A related object is to provide such systems and methods as permit improved traffic control without requiring additional signaling pins and without undue consumption of bandwidth or increases in latency.

SUMMARY OF THE INVENTION

The present invention has application, by way of example, in digital data systems in which messages pass along a link interconnect fabric from one node or device to another node or device. The nodes may be end points (such as processor or storage units), or may be intermediate devices or branch points (such as routers or switches in the interconnect fabric). Messages are packets having a defined format including, e.g., a header portion, typically with source and target addresses, and codes indicating message-type or other information, followed by one or more data or other fields.

In accordance with one aspect of the invention, a first node ("sending" node) of such a digital data system sends a data transmission comprising one or more message packets to a second node ("receiving" node) over a link of an interconnect fabric. The receiving node returns a control symbol to the sending node for each packet received on the link. The sender uses information in that symbol to control the further transmission of message packets to receiver over the link.

A system according to the invention ensures message efficiency and integrity on individual links of the fabric. Transmission errors occurring while the packets are transmitted over a plurality of links, e.g., from one end node to another, are corrected essentially when and where they occur. As a result, the endpoint devices are not tied up awaiting completion of a transaction or multi-packet transfer before they can detect errors or initiate recovery.

In accordance with further aspect of the invention, the second node of a digital data system as described above is connected to a further node (a third node, e.g., another endpoint) via a further link of the interconnect fabric. For each packet received from the first node, the second node returns a control symbol, e.g., acknowledging proper packet reception or indicating an error, for example, in the content or sequence of the received packet, before or while passing that packet onto the further node.

The first node of a digital data system as described above can respond to a control symbol indicating proper packet reception, for example, by clearing its buffers of the

packet(s) previously transmitted, validly received and acknowledged. Conversely, it can respond to a symbol indicating an error by retransmitting corrupted or lost packets.

Further aspects of the invention provide a digital data system as described above in which the control symbol includes status information indicating the availability of ports, processing resources or packet buffers in the second node. Such information can be used by the first node to regulate its transmission rate efficiently to the available receiving resources without deadlock.

In accordance with a further aspect of the invention, control symbols used to acknowledge received packets may be specially-delineated to optimize flow or integrity at the link level, i.e., between adjacent nodes and/or elements in the interconnect fabric. These acknowledgement control symbols may be transmitted as the packets are being received, without corrupting the packets. As noted elsewhere herein, the symbols may carry information effective to regulate message flow, identify faulty message data, and otherwise enhance the speed, accuracy or efficiency of communications over the link.

According to other aspects of the invention, the first and second nodes in a digital data system as described above include state machines that operate in accordance with commands or data contained in the control symbol to coordinate the handling of transmitted and received packets to assure packet integrity without unduly burdening message buffers.

The nodes in systems described above, moreover, can employ a variety of mechanisms to detect different types of errors. For example, a message packet transmitted by a sending node may comprise a header portion and a data portion, with at least the data portion including an error code. The receiving node uses that error code to at least detect, if not correct, errors in data or overall packet content.

In a related aspect, the sending node includes an "early" error code with the header portion of the packet. The receiving node can inspect that portion of an incoming packet to detect an error condition, returning a control symbol over the link even before the full packet

has arrived. Both the receiver and sender may discard post-error data so that both nodes back up to the last valid data and are immediately free to continue.

Still further aspects of the invention provide a digital data system as described above, wherein a part of the header of a message packet changes as the packet passes through the fabric and wherein other parts of the header (as well as the entirety of the data portion) may be invariant. The changeable part may include a sequence identifier, which each receiving node compares against an expected value to detect errors in transmission ordering. An error detection mechanism employed for the changeable part includes logic comparisons with internal format or external protocol attributes. An additional error code included in the packet and applicable to the invariant parts is checked as the message packet passes each node.

These and other aspects of the invention are evident in the drawings and the description that follows.

BRIEF DESCRIPTION OF THE DRAWINGS

Features and advantages of the invention and its structure and operation will be understood from the description and illustrative drawings herein, wherein:

Figure 1 illustrates a digital data system having a message passing system in accordance with the present invention;

Figures 1A - 1E illustrate various devices that may be connected as nodes or endpoints in the system of Figure 1;

Figure 2 illustrates two nodes connected by a link in a system of the invention;

Figure 3 illustrates representative formats for a message packet on the link of Figure 2;

Figures 4 - 4A illustrate embedding of a control symbol in a message packet;

Figures 4B – 4G illustrate the timing of framing signals that delineate the start of message packets and illustrate packet termination;

Figure 5 illustrates format of physical layer field for regulating flow of a packet transmission on a link in a system according to the invention;

Figure 6 illustrates error protection of fields in a control symbol for regulating flow of data on a link in a system according to the invention;

Figure 7 illustrates the physical layer fields added to a packet;

Figure 8 illustrates error coverage of initial field of a packet header;

Figure 9 illustrates control symbol error correction;

Figure 10 illustrates SECDED error protection of fields;

Figure 11A illustrates one example of a naturally 32-bit aligned packet of less than or equal to 80 bytes;

Figure 11B illustrates an example of a naturally 32-bit aligned packet of greater than 80 bytes;

Figure 11C illustrates an example of a padded 32-bit aligned packet of less than or equal to 80 bytes;

Figure 11D illustrates an example of a padded 32-bit aligned packet of greater than 80 bytes; and

Figures 12, 12A illustrate CRC computation.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

Architecture and Interconnect Fabric

Figure 1 illustrates generally an embodiment of a digital data system 100 employing link interconnect message communication in accordance with the present invention. The system includes a plurality of nodes, some of which are shown as digital devices 101, 102, 103, 104, . . . In the illustrated embodiment, these are processors with associated memories, though, they may also comprise other digital devices capable of generating, handling and/or passing message packets.

In this regard, it will be understood that messaging over interconnected links as described herein has wide application, e.g., to systems that implement direct memory access, to systems that provide communications between processors in a domain, as well as to systems that provide such communications between lower-level digital devices or components. Examples of the former are provided by Figures 1 and 1A. Examples of the latter are provided by Figures 1B and 1D (a stand-alone processor or processor/agent with no storage or I/O capability -- other than its interface to the link fabric); 1C (a stand-alone memory with no processor -- other than a controller that interfaces to the link fabric); and 1E (a switch processing element that receives multiple inputs along links in the interconnect fabric and generates corresponding output). Other devices or components in which such messaging has application include a bridge 115, as shown in Figure 1, or an I/O device that operates to move data into or out of local or remote memory or that interfaces static storage or other devices or systems.

The illustrated devices 101, 102, 103, 104, ..., are interconnected and communicate by a link fabric 110, which passes messages using buses, routers, switches, electrical, optical, electro-optical or other message-carrying elements. The routers and switches, by way of example, are vertices of the fabric. It will be appreciated that nodes 101, 102, 103, 104, . . ., may form vertices of the fabric as well. Conversely, it will be appreciated that the routers, switches, and certain other message-carrying elements (i.e., those capable of generating and/or handling message packets) may be nodes. Regardless of whether nodes or other

vertices, these devices may be embodied on a single piece of silicon, on a single board, in a single unit, or distributed among several interconnected boards and/or units.

In general, communications are effected in the illustrated system by messages which pass between desired endpoints in the system along links, each link extending between one vertex and another vertex. To this end, the nodes and/or other vertices may be associated with addresses. Thus, for example, illustrated digital devices 101, 102, 103, 104, . . . , each have an address or identifier by which they may be referred in message packets sent on the fabric. Messages are packets having a defined format including, e.g., a header portion with source and target addresses, and message-type or other codes, followed by one or more data or other fields, one example of which is described below in connection with Figure 3. The format of a message packet may differ for different types of communications or transactions, but is predetermined for each type of message.

In some embodiments of the invention nodes and vertices of the link interconnect fabric 110 may be constructed and operated in the manner of the crossbar switches described in U.S. Patent 5,598,568, as modified in accord with the teachings herein. The teachings of the aforementioned patent are incorporated herein by reference in their entirety.

In general, the architecture of a link fabric messaging system is intended to pass messages of defined formats and relatively small size at the link, or physical, level, using a defined protocol that allows the nodes at the ends of a link to dependably effect their message reception, handling and transmission at high speeds. The link-level message reception, handling and transmission may be effected with registers and timing or control elements, which may be implemented on a single piece of silicon or, otherwise, with a relatively small amount of logic. Such message handling circuitry may be fabricated integrally with a node or other vertex of the fabric (e.g., on a chip or board level), or it may be separate or free-standing device.

Figure 2 illustrates a link 125 interconnecting two nodes 120, 130 in a system of the present invention. The illustrated nodes are, by way of example, an endpoint (such as workstation 101) and a branch point (such as router or switch). The latter typically transfers

messages received on the link from node 120 further along the link interconnect fabric, and vice versa.

Each of illustrated nodes 120, 130 operates under a set of instructions such that, e.g., messages received at an input port of the node may be placed in an input section buffer, and messages or data for forming messages are passed for transmission via an output port.

The nodes may have plural input ports, or plural receiving buffers in the receiving section, for receiving messages from different links, or for receiving and ordering multiple packets constituting one or more than one different messages. The buffers may be configured so as to temporarily store received packets while message integrity and completeness are confirmed or corrected, or any necessary address, control or path configuration data are processed. Similarly, multiple output buffers and/or ports may be provided to buffer or handle data which is being transmitted (e.g., in order to match the node's I/O processing rates), to buffer data which has been transmitted (while awaiting confirmation of reception), or to otherwise hold data during a necessary message processing sequence.

Further details of one suitable embodiment of the nodes as described herein for carrying out such message communications may be found in the two United States Patent Applications entitled BUS PROTOCOL INDEPENDENT METHOD AND STRUCTURE FOR MANAGING TRANSACTION PRIORITY, ORDERING AND DEADLOCKS IN A MULTI-PROCESSING SYSTEM, and entitled METHOD AND APPARATUS FOR MAINTAINING PACKET ORDERING WITH ERROR RECOVERY AMONG MULTIPLE OUTSTANDING PACKETS BETWEEN TWO DEVICES, being filed of even date herewith and bearing attorney docket numbers SC11054TS-1 and SC11054TS-2, respectively. Further details regarding the use of control symbols, or short link level commands or control communications transmitted between nodes on a link may be found in the United States Patent Application entitled DIGITAL DATA SYSTEM WITH LINK LEVEL MESSAGE FLOW CONTROL, also being filed of even date herewith and bearing attorney docket number 102323-61. Each of those patent applications is hereby incorporated herein by reference in its entirety.

Briefly, in a link fabric interconnect, processing elements such as switches provide the interconnect fabric between end points; a single link (e.g., a bus or other electrical, optical, electro-optical, or other signal carrying element) connects to a node at each end. The nodes utilize that link to transfer, not only message packets, but also requests and responses for link-level information among each other. The link-level information can include information such as an indication of buffer status, an ACKNOWLEDGMENT of message receipt, or other information useful at the link level to assure efficient message flow and message integrity. The nodes follow defined protocols to supply or request such information or necessary parameters, and to acknowledge receipt of, or to request retransmission of messages in progress, such that the messaging traffic proceeds smoothly. Thus, flow control and error handling are managed between each electrically connected device pair rather than between the original source and final target of the transaction.

The links may be full duplex, and the nodes configured to send and receive message packets. Illustratively herein each packet is composed of a number of 32-bit words (symbols), and includes an error code such as a CRC code. The words of a packet are sent over the link, and a node receiving the symbols at the other end of the link checks the error code, and if the code indicates an error occurred in transmission, the receiving node sends a request on the link for retransmission. If the error code indicates the message is correct, the receiving node sends an ACKNOWLEDGMENT that it properly received the message. Once the receiving node has acknowledged successful message reception, the sending node may clear its buffers of the transmitted message. Thus, the nodes at each end of a link are configured to assure that messages are faithfully passed over the link, yet may operate with relatively small buffer capacity to check, confirm and pass along messages in order. Operation of the nodes to efficiently detect corrupt packets and carry out retransmission between such interconnected nodes is described in detail in the aforesaid patent application entitled METHOD AND APPARATUS FOR MAINTAINING PACKET ORDERING WITH ERROR RECOVERY AMONG MULTIPLE OUTSTANDING PACKETS BETWEEN TWO DEVICES, attorney docket number SC11054TS-2.

The link-level information, that is the control information (distinct from the message packet information) passing between two adjacent nodes, may be encoded in relatively short

blocks. In accordance with one illustrative embodiment, each of these control instructions is encoded in a single word, i.e. a "control symbol". A control symbol may be sent by a node that is transmitting a message to a receiving node. A control symbol may also be sent by a node that is receiving a message to the node that is transmitting that message. Control symbols may also be sent over the node when no messages are being sent. Two nodes connected to the link (e.g., two switch nodes, or a switch and an endpoint node) may use the control symbols to exchange link-level information for controlling various link-level mechanisms. Preferably the system is implemented such that a link-level control symbol never travels over more than one link. In one typical situation, a control symbol may be transferred during the period between two message packets. It may also be sent during or within a message packet transmission as described in the above-reference patent application entitled DIGITAL DATA SYSTEM WITH LINK LEVEL MESSAGE FLOW CONTROL, attorney docket 102323-61. These symbols are referred to herein as embedded control symbols.

Operation of the error detection and correction mechanisms of the present invention will be understood in the context of a link fabric messaging system described below. Such a system may advantageously be effected such that a message packet sent by one node on a link to a receiving node at the other end of the link is comprised of a sequence of symbols (e.g., 32-bit words) that are transmitted in order over the link. A frame signal may be asserted at the start of the packet to mark the first symbol of the packet (e.g.; the first 32-bit word), and the messages formatted, and handled within each node, such that the subsequent symbols of a packet are aligned with word boundaries established by the frame signal.

By way of physical example, a link communication chip node may employ an input or output port that is eight or sixteen bits wide in order to accept or transmit four-byte words. Alignment of the transmitted symbols on word boundaries allows the node, e.g., the chip, to operate internally with a slow (e.g., divided-down) clock, and provides a dependable mechanism for alignment and processing of formatted symbols and messages exchanged through the ports between nodes on the link. When so aligned, particular bit positions of defined fields are detected and processed to determine message types, administrative or status

information or commands for node operation, and transmission errors. Suitable methods of transmitting and aligning data are described below with reference to Figure 4.

In a link fabric messaging system of this type, a message packet may defined for each of various transactions or communications between devices. Thus, for example, the system may utilize messages of one format for direct writes to memory; messages of another format for querying a remote device for status; messages of still another format for device configuration or other data; and so forth. Each message typically has a header and may have a data or other subsequent portion. The header can indicate message origin, destination, message type, size of data portion or other subsequent portion. The header may also include one or more error detection or correction codes, a priority code to govern message handling, and/or information that the receiving node may use for configuration or routing.

In general, a message received by a node may be placed in an input buffer, appropriate bit positions inspected for necessary information, particular bits or fields may be modified, and may be passed for transmission along another, or back along the same link, as appropriate to the message type. A node may take an unspecified period of time between receiving a packet and transmitting it to the next node to allow for in-transit housekeeping, such as error checking and correction, coordination when words are received out of order in a multi-word message packet, setting up of further buffers, sending of an acknowledgment or a request for retransmission, and other such operations. Aspects of the invention provide systems as described above that utilize the aforementioned control symbols to coordinate intra-link communication between sending and receiving nodes during these operations, and to detect and/or correct errors in transmission on the link.

In one embodiment of a message-passing link interconnection of the present invention, the links transfer packets in order, and each packet has a field *ackID* indicating this order. A brief discussion of ordering or correction mechanisms related to this field and operating at the link level to enhance message flow are included here, and are more fully described in the aforesaid METHOD AND APPARATUS FOR MAINTAINING PACKET ORDERING patent application. The *ackID* field may be reassigned, and thus may be different, each time the message passes on a different link, so a special error code covering

only the initial bits of the symbol may be used to check header symbol integrity without having to compute an error code for the entire massage or packet. Alternatively, these bits may be checked by simple logic and excluded from the portion to which a packet error code is applied.

The output section of each node may, for example, maintain a three bit counter, and set the *ackID* of each outgoing packet, starting at *ackID* 000 for the first packet after reset. The counter is incremented each time a packet is sent, and the counter rolls over after 111. Each input section may then maintain an input counter that indicates the next expected packet number, and checks that each received packet has the correct sequence number. If no errors are detected and the *ackID* is the expected number, the receiving node may sends a control symbol (denoted the PACKET_ACCEPTED ACKNOWLEDGE control symbol) over the link to the adjacent (sending) node acknowledging proper reception. Otherwise the receiving node sends a symbol (denoted the PACKET_NOT_ACCEPTED control symbol) to the adjacent node causing it to initiate a retry mechanism. Both of these acknowledge control symbols include the *ackID* of the received packet. This flow control mechanism assures that transmitted messages are faithfully received, corrected if needed, and passed along the link fabric, thus allowing a transmitting node to quickly clear its buffers of message transactions, so that a receiving node quickly receives and passes along complete packets, while avoiding the corruption of packets in transit.

The implementation of error management by communicating control symbols between adjacent nodes will be best understood in the context of a specific messaging format and protocol, which will be discussed below to illustrate representative implementations of the control and coordination of messaging on a link.

By way of overview, messaging is effected as follows. Each direction of a link is composed of data signals, a frame signal, and a clock. Various physical implementations are possible. The frame signal helps delineate message packets, e.g., so they may be aligned and processed in a receiving node. The frame signal is asserted at the beginning of a message packet. The frame signal may also be asserted at the beginning of a control symbol. All packets and control symbols, whether sent over 8-bit or 16-bit ports, are aligned to 32-bit

word boundaries. Packets that are not naturally aligned to a 32-bit boundary are padded. Control symbols may be nominally 16-bit quantities, but are preferably defined as a 16-bit control symbol followed by a bit-wise inverted copy of itself, so that it aligns to the 32-bit boundary. The presence of a bit-wise inverted copy may be applied as an error detection mechanism to the interface. These 32-bit quantities will be referred to as aligned control symbols.

A single bit position, termed the "S" bit, may be used as a flag to encode whether the symbol is a control symbol (*S*= 1) or a message packet header (*S*= 0). The receiving node passes an incoming message in proper alignment through registers or logic and inspects its *S* bit values to determine whether it is a message packet or control symbol, and to effect its handling. The protocol for sending message packets on a link includes acknowledging each received packet. A link requires an identifier to uniquely identify an acknowledgment. This identifier, known as the acknowledge ID (or *ackID*), is a three bit field in the packet header, allowing for a range of one to eight outstanding unacknowledged request or response packets between connected nodes. The *ackID*s are assigned sequentially to indicate the order of the packet transmission. The acknowledgments themselves are each one of a number of different aligned control symbols, discussed separately below, and operate to regulate flow of messages efficiently, and to implement retransmission of packets that were not received or were found to have errors, to ensure that valid packets are received in order and the sending node may clear its packet transmission buffers.

Message Packet Format

Figure 3 illustrates representative forms of a message packet useful in one embodiment of a message-passing link of the invention such as link 125 of Figure 2.

The word or the basic unit (not necessarily a control symbol, discussed elsewhere herein) is a four-byte word. In various physical implementations, the bytes may, for example, enter each node via a port eight bits wide, sixteen bits wide, or otherwise. A packet is composed of a number of such words, e.g., a sequence or ordered set of words, for example four, five eight or more words. It will be understood that higher level messages such as data

transfers may be transmitted as message packets having a greater number of words, may be transmitted as a plurality of packets, or both. The templates are illustrative only, and particular packet types may be configured differently for different types of messages, such as transaction messages, streaming data messages, requests, responses, or other types of messages. Furthermore, link interconnect systems of the present invention may employ a basic message structure utilizing larger (or smaller) words to form the messages. Thus, while the illustrated templates are shown with a basic four byte symbol, other systems may employ sixteen-bit, thirty-two bit or sixty-four bit interfaces, and may utilize formatted basic units-packet templates and symbols, with halfwords or doublewords of correspondingly different size.

In this example, a four-byte word or symbol is the basic unit for messaging on a link. Each type of message packet has a defined format, with a header comprising a specific set of fields, and these are passed along links to an endpoint. A single-word control symbol is used for communications between nodes on a link, and these are both short and highly formatted. The control symbols, passing between two adjacent nodes at the ends of a single link, do not require addresses, and a small number of bits suffice to encode the link-level control actions (discussed further below) indicated by the control symbol. In other cases, where a symbol starts or continues the header or is subsequent portion of a message packet, the form may differ, and may include specific fields for a header, or simply data in the case of words constituting the data portion of a message packet.

Packet Start and Symbol Delineation, Packet Termination

In the illustrated embodiment, the input and output port of each node is either one or two bytes wide. Like protocols are used for both the 8- and 16-bit wide versions, the only difference being the number of pins used to transmit the packets and aligned control symbols.

At the outset, it will be appreciated that alignment of message or control symbol words in an illustrative physical implementation of a link may be effected as shown in Figure 4. In a message packet, the first (header) symbol of a packet is asserted with a frame signal ***F*** (e.g., a line high, or line to zero signal). A node, such as node 130 in Figure 2, receiving a

symbol when the frame signal is asserted, aligns the symbol and processes (e.g., with logic) defined bit positions in the first symbol to detect bits encoding information such as specifying a packet type, priority level, source and destination address, and checking or other parameters that determine packet handling. Following the first symbol, the successive symbols of a packet header if any are aligned on the 32 bit word boundaries, and have defined fields that may vary for each type of message packet. Such fields may specify remote device storage addresses to be read or be written to, information about the type of transaction, block size parameters or other such information. Packets that carry data may have data fields.

By way of digression, figure 4A illustrates such embedding of a control symbol C in an ongoing message packet transmission. As shown, the bytes of the symbol C are inserted at a 32-bit symbol boundary of a stream of bytes constituting the initial symbols P1 of a packet. The frame signal ***F*** toggles at the start of the control symbol C. The remaining symbols P2 of the message then follow immediately after the four-byte control symbol, without additional frame or other indication. The receiving node is configured to process the control symbol C separately, while placing P1 and P2 in aligned order so that they are checked or handled and passed along the link fabric as a complete and properly-formatted and ordered message packet.

The frame signal ***F*** used to delineate the start of a packet or a control symbol on the physical port is a no-return-to-zero, or NRZ, signal. This frame signal is toggled for the first symbol of each packet and for the first symbol of each aligned control symbol. Therefore, if a 16-bit symbol contains a logical packet format type or a control symbol field, the frame signal must toggle. In order for the receiving processing element to sample the data and frame signals, a data reference signal is supplied that toggles on all possible transitions of the interface pins. This type of data reference signal is also known as a double-data-rate clock. These received clocks on devices with multiple ports have no required frequency or phase relationship. The data reference signal is always rising on the 32-bit boundary when it is legal for the frame signal to toggle as shown in 4B – 4E.

The frame signal ***F*** is not toggled for other symbols such as remaining packet header and data bytes. However, it is toggled for all idle symbols between packets. This means that

the maximum toggle rate of the control framing signal is every 4 bytes, and that the framing signal is only allowed to toggle on every fourth byte, meaning the framing signal is aligned to a 32-bit boundary as are all of the packets and aligned control symbols. Additionally, the data reference signal must transition from low to high on this same boundary. Examples of these constraints are shown in Figure 4B and Figure 4D for an 8-bit port and Figure 4C and Figure 4E for a 16-bit port.

Errors on the framing and data reference signals can be detected either directly by verifying that the signals transition only when they are allowed and expected to transition, or indirectly by depending upon detection of packet header or CRC or control symbol corruption if these signals behave improperly. Either method of error detection on the framing and data reference signals allows error recovery by following the mechanisms described below.

A packet is terminated in one of two ways: the beginning of a new packet signals the end of a previous packet; or the end of a packet may be marked with one of the following 16-bit symbols: an aligned end-of-packet (eop), restart-from-retry, link-request, or stomp control symbol. A stomp symbol (as described in aforesaid United States Patent Application entitled DIGITAL DATA SYSTEM WITH LINK LEVEL MESSAGE FLOW CONTROL, being filed of even date herewith and bearing attorney docket number 102323-61) is used if a transmitting processing element detects a problem with the transmission of a packet. A device may choose to abort the packet by sending the stomp symbol instead of aborting it in a different, possibly system fatal, fashion-like corrupting the CRC value. The use of such control symbols passed between adjacent nodes allows faulty or partial transmissions to be quickly caught, e.g., stopped from further propagation and discarded from message-handling buffers in affected nodes

The restart-from-retry can abort the current packet as well as be transmitted on an idle link. This symbol is used to enable the receiver to start accepting packets after the receiver has retried a packet.

The link-request symbol can abort the current packet as well as be transmitted on an idle link and has several applications. It can be used by software for system observation and maintenance, and it can be used by software or hardware to enable the receiver to start accepting packets after the receiver has refused a packet due to a transmission error.

A receiver must drop a stomped packet without generating any errors and must then respond with a retry acknowledgment symbol unless an acknowledgment has already been sent for that packet or the receiver is stopped due to an earlier retry or error. If the receiver is not already stopped it does not stop due to the stomp. Figure 4F is an example of a new packet marking the end of a packet. Figure 4G is an example of an aligned end-of-packet symbol marking the end of a packet. The stomp, link-request, and restart-from-retry cases look similar.

Error Management

In accordance with a principal aspect of the present invention, error mechanisms detect corrupted or congested data at the individual physical links of the system, and operate to correct and control message flow. For this purpose certain initial fields in the packet header and in the control symbols are used. These are physical layer fields - i.e., fields that are detected and processed, e.g., by logic or circuitry, at each node for effecting the physical message handling and control at the node. These include the single bit *S* flag, that indicates to the link level data handling unit whether the word is a control symbol (*S*= 1) or a request or response message packet header (*S*= 0). Other physical layer fields include the 3-bit request or response packet identifier *ackID* that is used to identify a packet when communicating an acknowledgment back to the sender of the packet, and a 2-bit priority code *prio* that is used to determine message handling priority (as described in the aforesaid United States Patent Application entitled BUS PROTOCOL INDEPENDENT METHOD AND STRUCTURE FOR MANAGING TRANSACTION PRIORITY, ORDERING AND DEADLOCKS IN A MULTI-PROCESSING SYSTEM, attorney docket number SC11054TS-1).

A control symbol includes a short e.g. 3-bit field (denoted *stype* below) encoding the command, query or link data that is specified by the control symbol. In different

embodiments and implementations, the control symbols or packet headers may include other fields, examples of which are discussed as they arise, below. It will further be understood that a given field may encode different information depending on its context, e.g., in a flow control symbol certain field bits may specify receiver buffer status, whereas the same bit positions in a different control symbol may identify a cause of a detected error. The handling of a received message may be governed by a state machine at the input port of a node, that inspects bit positions and undertakes processing steps governed by to carry out the functions described herein.

These steps include acknowledgment messages, communicated by short control symbols that, in turn, cause the upstream node receiving the symbol to undertake appropriate action for the reported reception, or mis-reception indicated by the control symbol. Because receipt of an acknowledgment control symbol does not imply the end of a packet, these symbols can be sent as embedded control symbols within a packet. They may also be sent when the link interconnect between the two nodes is otherwise idle.

Appendix A attached hereto illustrates details of the message handling and communication steps undertaken by the sending and the receiving nodes in response to various error conditions. The link level mechanisms of in transit packet evaluation and correction of corrupted transmissions or recoverable errors will be better understood from a discussion of the format and content of the message packet headers, and the fields employed in the acknowledgment control symbols and the corresponding operations induced at a node sending or receiving such a control symbol as discussed below.

In accordance with a first aspect of this feature of the invention, a node receiving a message packet may send a PACKET_ACCEPTED ACKNOWLEDGMENT control symbol to the upstream node to indicate that the receiving node has properly received the entire packet and has taken responsibility for sending the packet along to its final destination. This symbol may be generated at any time after the entire packet has been received, and has been found (e.g., by applying a packet CRC or other error checking code, or by applying parsing rules) to be free of transmission errors. The upstream or connected node that originally sent the packet may then, upon receiving the PACKET_ACCEPTED ACKNOWLEDGMENT,

reallocate its resources, for example, by releasing the packet assembly and transmission buffers that had been allocated to the subject packet.

Systems of the present invention employ message packets having headers and control symbols having formats wherein an initial portion of the header or the control symbol format includes physical layer fields, i.e., fields that are recognized e.g., by logic or simple circuitry in each node, for affecting the physical message handling and control at the node. Thus, for example a message packet header format may include the *S* flag, the 3 bit *ackID* field and a 2 bit priority field. Use of a priority code for assuring message flow is discussed in great detail in the aforesaid BUS PROTOCOL INDEPENDENT METHOD patent application. Control symbols, short formatted command or control type words passed between adjacent nodes, may include the *S* flag, *ackID*, a buffer status field to specify the number of available buffers in the connecting node, and a symbol type indicator identifying a command or action which the control symbol implements. A short error code field, such as a 5-bit single error correction/double error detection (SECDED) code may be included as a check or double check of some or all of the initial (header) bits.

The 3 bit *ackID* field of a transmission is assigned by the sending node, starting at zero after a reset and following the sequential procession, 1, 2, 3, 4, 5, 6, 7, 0, 1, etc. Thus, once a transmission is initiated, the receiving node knows what *ackID* number it is expecting at each stage. Packets are only accepted by the receiving node in the sequential order specified by the *ackID*, and the receiving node may indicate proper receipt by returning a PACKET_ACCEPTED control symbol back to the sending node. This ordering allows a receiving device to detect when a packet has been lost by internal checks, and also provides a mechanism for maintaining ordering.

Figure 5b illustrates a suitable format for such a PACKET_ACCEPTED ACKNOWLEDGMENT control symbol. It includes the control symbol *S* bit *S*=1, the 3-bit acknowledgment for the received packet *ackID* that had been assigned by the original sending node (here referred to as the target *ackID*, since the upstream sending node is now the target of the PACKET ACCEPTED AKNOWLEDGMENT control symbol), a 4-bit buffer status field (*buf_status*) and a 3-bit control symbol type indicator (*stype*). A 5-bit single error

correction/double error detection (SECDED) error code is also provided, which allows the upstream node to check for corruption of the acknowledgment control symbol.

A receiving node may also "retry" a packet, by returning a PACKET_RETRY acknowledgment control symbol to the sender. This control symbol signals the sender to transmit the affected packet again. The PACKET_RETRY acknowledgment control symbol may be sent in response to some temporary internal condition of the receiving node, such as having all its message handling buffers full or having its processor or handling logic fully occupied with another transaction. The receiving device then silently discards all new incoming packets until it receives a control symbol from the sender indicating it should restart from the retry point. The sender then retransmits all packets starting from the retried *ackID*, reestablishing the proper ordering between the devices. The packet sent with the retried *ackID* may be the original retried packet, or may be a higher priority packet (if one is available for transmission), as discussed in the aforesaid BUS PROTOCOL INDEPENDENT METHOD patent application, thus allowing higher priority packets to bypass lower priority packets across the link and avoid deadlock.

Thus, the PACKET_RETRY acknowledgment control symbol allows an intervening link in the fabric to reconstitute an interrupted packet, without discarding and repeating the full transaction, and without waiting for message assembly and retransmission requests from an end point device to which the transaction was addressed. Further details of operation are set forth in the flow charts of Appendix A, filed herewith.

Embodiments of the invention further include error detection and correction mechanisms to guard against corruption or loss of data. These may include error detection or correction codes both for the initial header information for a control symbol or a message packet, and for the remaining data in the case of message packets.

In accordance with further aspects of the invention, if a receiving node encounters an error condition it may send a PACKET_NOT_ACCEPTED control symbol. This control symbol preferably includes an error-type code indicating the error condition that caused non-acceptance. Once a faulty packet has been identified, the receiving node discards all new

incoming packets until retry is effected. This assures that packet order of the transmission is maintained. If the error condition is due to a transmission error, the node may apply error recovery mechanisms as described further below.

Adjacent nodes may use control symbols for operations such as reserving buffer space for different priority packets to be sent. The buffer status field of the PACKET_RETRY control symbol serves to notify the sender of available buffer space. When this information is passed between adjacent nodes on a link, a sender may implement a transmission protocol to stop transmitting packets when no receiving packet buffer is available, or stop until a high priority packet requires transmission, or until sufficient buffer space is freed to resume packet transmission.

Different embodiments may be implemented with only a small amount of buffering available; in that case the node may be set up to operate like an input FIFO, or data stream buffer. Such an embodiment may have insufficient buffering for even one large packet, and may depend upon cyclic buffering, i.e., resource reuse, to send and receive large packets. In that case, the buffer status field may simply indicate the maximum number of buffers, and the node may rely on other flow control mechanisms to manage the interface packet stream.

Operation of these message handling steps and detection or correction of errors are made more efficient by a combination of error protection mechanisms that operate successively and in coordination with the packet transmissions.

In accordance with one embodiment of the error detection mechanism of the present invention, packets sent to a node employ one or more CRC codes, and preferably also apply a simple error correct/detect code (e.g., a SECDED) to provide a combination of enhanced integrity and speed of processing. This combination of different error checks allows errors to be detected or corrected at the link level with minimal impact on data transfer speed. A preferred embodiment applies a CRC to portions of a packet that do not change as the packet passes through the link fabric. These may be, for example, portions of the packet outside the physical layer/ header fields. A simple error code may then be applied to one or more changing fields within the header or initial bits. This avoids having the interconnect fabric

regenerate a CRC value as the packet moves through the fabric, thus avoiding a time consuming and computationally intensive step at each link.

Other embodiments may alternatively, rather than applying a simple SECDED or similar error code, detect transmission errors (such as in control symbols) in hardware, for example as inconsistencies or violations of the control symbol format or protocol. This is possible since control symbols are heavily coded and any out of place bits will result in detection of an undefined state or error condition.

Figure 5 shows by way of example, a format for the physical layer fields of a request or response packet header in line (a), and shows in line (b) a format for the physical layer fields of a control symbol. These physical layer fields of the packet, line (a) are preferably prepended to other logical fields in the packet, as indicated in line (c). This produces a packet wherein initial header bits relate to processing or operation of transmission on the link, while later bits relate to packet contents or aspects of packet handling.

The aforesaid United States Patent Application for METHOD AND APPARATUS FOR MAINTAINING PACKET ORDERING WITH ERROR RECOVERY AMONG MULTIPLE OUTSTANDING PACKETS BETWEEN TWO DEVICES describes operation of two connected devices using retry protocols to correct a received packet when it is received out of order (e.g., with an unexpected *ackID* as described above) or is found by an error detection module operating in a receiver to have an error. The present invention combines multiple different error detection mechanisms to portions of packets or control symbols to more effectively detect substantially all errors and provide continuous but highly efficient error protection as a packet moves through the interconnect fabric. The use of different error protection mechanisms for different portions of a message allows changeable parts of the message to be protected efficiently on each link, while avoiding re-calculation of computationally burdensome error protection codes (such as a CRC) on major portions of the message as it moves trough the link fabric.

The control symbols may be protected in two ways: i) the *ackID* and *stype* fields may be protected by a single error correct, double error detect (SECDED) code, discussed further

below in connection with Figure 10, below; and ii) the entire control symbol may be protected by the bit-wise inversion of the symbol used to align it to the 32-bit word boundary described above. This format redundancy allows precise error correction and detection.

When providing a SECDED for error coverage of the control symbols, it is not necessary to cover all bits. The *S* bit and the *buf_status* field need not be covered with the SECDED code. This is because a failure in the *S* bit will cause the control symbol to be interpreted as a packet, which will therefore either be rejected due to an unexpected *ackID* value, or will generate a CRC error when a CRC code is applied to the complete packet transmission. The buffer status field also need not be protected by the error correction mechanism because this field is present in most control symbols, including idle symbols discussed in the foregoing patent application (docket 102323-61). Further, this field is protected by the bit-wise inversion in the second half of the symbol. An error in this field need not be treated as an error condition because it is always an informative field that is not critical for proper system behavior. It serves to inform an adjacent node of the available number of packet buffers in the sending node. Thus, for example, if a corrupt value of *buf_status* occurs, a low value may temporarily prevent a packet from being issued, or a high value may result in a packet being issued when it should not have been, causing temporary overload or congestion, and resulting in a retry. In either case the problems are temporary and will properly resolve themselves through the protocol for transmission pacing or flow control.

Figure 6 illustrates such error protection coverage for a typical control symbol. As shown, the physical layer field other than the *S* and *buf_status* fields are covered by SECDED error protection.

Message packets as described above are preferably protected with a CRC code that also covers the two bit message-handling priority code field described above. The *S* and *ackID* fields need not be protected with an SECDED code (as with the control symbols), but instead are protected by protocol as described below. The reserved bits in a packet are assigned to logic 0 when the packet is generated, and are ignored when a packet is received. The reserved field is added to the combined flow control, transport, and logical packet as shown in Figure 7. The two reserved bits adjacent to the *prio* field are also protected with the

CRC. The new reserved field is unshaded, the related fields are lightly shaded, and the unrelated preceding and following bits of the transmission are illustrated as heavily shaded.

Figure 8 shows the error coverage for the first 16 bits of a packet header. CRC protects the *prio*, *tt*, and *ftype* fields and two of the reserved bits. (The *tt* field specifies certain system specifics, such as whether device IDs are 8- or 16- bit, and the *ftype* identifies the packet format type.) There is for each new packet an expected value for the *ackID* field at the receiver, bit errors on this field are easily detected and a error in this field causes the packet to be not accepted due to the unexpected value. An error on the *S* bit causes the packet to be misinterpreted as a control symbol, which is statistically likely to assure detection of an error state, either because the control symbols are very heavily encoded or because of an induced protocol violation such as an unexpected *ackID* value. Errors on the two unprotected reserved bits can be ignored.

Thus, all of the necessary initial information appearing in a control symbol or a packet header, or all of the changing fields that may vary from link to link, is protected against errors. The CRC that covers the major portion of a packet does not need regenerate a packet CRC value each time the uncovered physical layer fields are assigned as messages pass along the links of the interconnect.

It should be observed that even with these error detection mechanisms, and with the retry and correction mechanisms described in the aforesaid PACKET ORDERING patent application, some types of errors, such as a lost request or response packet or a lost acknowledgment, result in a system with hung resources. Preferably, to detect this type of error, time-out counters are provided that expire when sufficient time has elapsed without receiving the expected response from the system. Because the expiration of one of these timers should indicate to the system that there is a problem, this time interval should be set long enough to avoid signaling a false time out. The response to this error condition may be system dependent.

For example, an implementation may have one-time out counter for the logical layer and another for the physical layer. The logical layer time out occurs between the issue of a request transaction and the receipt of the response. The time-out interval for this packet pair is likely to be fairly long. The physical layer time out occurs between the transmission of a packet and the receipt of an acknowledgment symbol. This time-out interval is likely to be comparatively short because the packet and acknowledge pair only has to traverse a single link. For the purposes of error recovery in the case of a time-out on an acknowledge control symbol the time-out should be treated as (e.g., the node responds as though it were) an unexpected acknowledge control symbol rather than a corrupt control symbol. Such time-out registers may be specified depending upon a number of factors such as the link clock rate, the internal clock rate of the device, and the desired system behavior in an implementation dependent manner.

Error detection and correction is preferably done at the input port, and all recovery also initiated at the input port. Error detection can be done in a number of ways and at differing levels of complexity and robustness depending upon the requirements and implementation of a device. For example, a device may be implemented that ignores the SECDED capability on received control symbols and treats all potentially correctable errors as not correctable, resulting in a simple but lower performance design. Another device may be implemented to ignore the redundant half of an aligned control symbol and use the SECDED code as the only means of error detection and correction, resulting in a simple but less robust design.

Control symbols can recover silently from single bit errors on an aligned control symbol, described in Figure 9A, and from a number of multiple bit errors, described in Figure 9B. The column labeled "bit mismatch" of that Figure refers to comparing the two redundant halves of an aligned control symbol.

Attached hereto as Appendix A are SECDED error tables for the SECDED code defined in Figure 10, as well as six charts, numbered Figure A-1 to A-6 showing details of the initialization and the operation of sending and receiving nodes for message handling and error recovery steps. Two kinds of errors are detected at an input port: an error on a packet and an error on a control symbol.

Three types of packet errors exist: a packet with an unexpected *ackID* value, a corrupted packet indicated by a bad CRC value, or a packet that overruns some defined boundary such as the maximum data payload or a transactional boundary defined for its packet type. A node processing element that detects a packet error immediately transitions into an "input stopped due to transmission error" state and silently discards all new packets until it receives a restart-from-error control symbol from the sender. The device also sends a packet-not-accepted control symbol with the received *ackID* value back to the sender. The sender then initiates recovery (described below) for unexpected control symbols.

The two types of control symbol errors are i) an uncorrupted packet-accepted, packet-retry, or packet-not-accepted control symbol (which is either unsolicited or has an unexpected *ackID* value) or ii) a corrupt control symbol. The first case, an uncorrupted protocol violating acknowledgment (due to the unexpected *ackID* value), causes the receiving device to enter an "output stopped due to transmission error" state, immediately stop transmitting new packets, and issue a restart-from-error control symbol. The restart-from-error control symbol receives a response from the downstream node containing pertinent receiver internal state, including the expected *ackID*. This expected *ackID* indicates to the sender where to begin re-transmission because the interface may have gotten out of sequence. The sender then backs up to the appropriate unaccepted packet and begins re-transmission.

For example, the sender may transmit packets labeled *ackID* 2, 3, 4, and 5, and receive acknowledges for packets 2 and 4, indicating a probable error associated with *ackID* 3. The sender then stops transmitting new packets and sends a link-request for link status control symbol to the receiver. The receiver then returns a link-response control symbol indicating which packets it has received properly. The following are the possible responses and the sender's resulting behavior:

expecting *ackID* = 3 - sender must re-transmit packets 3, 4, and 5

expecting *ackID* = 4 - sender must re-transmit packets 4 and 5

expecting *ackID* = 5 - sender must re-transmit packet 5

expecting *ackID* = 6 - receiver got all packets, resume operation

expecting *ackID* = anything else - fatal (non-recoverable) error

The second case (namely, a corrupt control symbol) again causes the receiver to enter the "input stopped due to transmission error" state and send a packet-not-accepted control symbol with an unexpected *ackID* value to the sender. This informs the sending device that a transmission error has occurred and it will enter the recovery process described in the first control symbol error case described above. A special case occurs with a corrupt embedded control symbol. (Embedded control symbols, i.e., those sent within a message packet, are described in the aforesaid LEVEL MESSAGE FLOW CONTROL patent application.) In this case the packet-not-accepted control symbol contains the *ackID* value of the embedding packet, and the packet is discarded.

As noted above a SECDED code may protect initial parts (e.g., physical layer fields) of a message. Table A of Figure 10 contains a suitable single error correct, double error detect (SECDED) code for the physical layer fields and the control symbols. Data bits marked in the table are exclusive-OR'ed together to generate the protection bits. These bits correspond to the covered *ackID* and *stype* fields as shown in Table B of Figure 10.

The major portion of a message packet is preferably protected by a cyclic redundancy code (CRC). In a prototype embodiment, a 16-bit CRC was selected as the preferred method of error detection for the end point physical layer. This CRC is generated over all of a packet header and all of the data payload except the first 6 bits of the added physical layer fields as shown above. The initial value of the CRC is 0xFFFF, or all logic 1s. For purposes of simplifying CRC calculation, the uncovered 6 bits are assumed to be logic 0s. This checksum is appended to a packet in one of two ways. For a packet that has up to 80 bytes of header (including all logical, transport, and physical link protocol fields) and logical data payload, a single CRC value is appended to the packet.

For packets with greater than 80 bytes of header and logical data payload, one CRC value is inserted after the first 80 bytes, aligning it to the first half of the 32-bit word

alignment boundary, and a second CRC value is appended at the end of the packet. The second CRC value is a continuation of the first and included in the running calculation. That is, the running CRC value is not re-initialized after it is inserted after the first 80 bytes of the packet. This allows intervening devices to regard the embedded first CRC value as 2 bytes of packet payload for CRC checking purposes.

Advantageously, the embedded CRC value is itself used in the running CRC. As a result, from the CRC generator's point of view the running CRC value is guaranteed to be all logic 0's because the running CRC is XOR'ed with itself. This property is expected to be useful in an implementation.

The first or early CRC value can be used by the receiving end point to validate the header of a large packet, and to start processing the received data before the entire packet has been received. This frees up resources earlier and reduces transaction completion latency. If the final appended CRC value does not cause the total packet to align to the 32-bit boundary, a 2 byte pad of all logic 0s is postpended to the packet. Such pad of logic 0s allows the CRC check to always be done at the 32-bit boundary.

Switch devices in the link fabric must maintain the packet error coverage internally in order to preserve the integrity of the packets though the fabric. This will prevent device internal errors such as SRAM bit errors from silently corrupting the system. The simplest method for preserving error coverage is to pass the CRC values through the switch as part of the packet.

Figure 11A illustrates one example of a naturally 32-bit aligned packet of less than or equal to 80 bytes. Figure 11B illustrates an example of a naturally 32-bit aligned packet of greater than 80 bytes. Figure 11C illustrates an example of a padded 32-bit aligned packet of less than or equal to 80 bytes. Figure 11D illustrates an example of a padded 32-bit aligned packet of greater than 80 bytes.

One suitable CRC is given by the CCITT polynomial $X^{16}+X^{12}+X^5+1$. Figure 12 illustrates, by way of example, a 16-bit wide parallel calculation for this polynomial. Equivalent implementations of other widths can be employed. In the Figure, C00 - C15 indicate contents of the new check symbol, e00 - e15 indicate contents of intermediate value symbols (e.g., successive steps in the computation e00 =d00 XOR c00, e01 = d01 XOR c01, ...), where d00 -d15 are the contents of the next 16 bits of the packet, and c00 - c15 are the contents of the previous check symbol for the pipeline shown in Figure 12A.

Hop Count

It will be understood that systems of the present invention may form a network, and are particularly advantageous for implementation of networks wherein processors , devices or directly accessed memory may be distributed and addressable in the network. Formats of particular message packets may be configured to include address or extended address fields for efficiently carrying out direct writes, reads or other functions to such memory, processor or other devices. Among the transactions that may be effected are maintenance or administrative type transactions, such as querying a control and status register of a node to determine its availability for receiving message packets or to configure routing of such packets.

In many instances, the intermediate nodes of a link fabric network, such as switches, need not have a device ID, yet it may still be desirable to address such nodes, for example to determine their availability, or to set up or reconfigure a route between end point nodes for a data transmission.

In this case, control messages for effecting such queries or transactions may be sent to a node by including a "hop count" field in the control message. As this message passes along the link fabric, each node inspects the hop count field, and if it is non-zero, decrements the hop count by one, and passes the control message to the next link. When the hop count field is zero, a receiving node then reads and implements the command or transaction indicated by the message content. Thus, intermediate nodes may be addressed in the manner of a token

passing ring, for effecting such transactions as reading registers, determining buffer status, reserving ports and other administrative or handshake functions.

This manner of addressing intermediate nodes of the link fabric allows much of the administration of link level communications to be carried on without having to configure hardware for an excessively large number of device IDs.

The invention being thus disclosed and illustrative embodiments described herein, further variations, modifications and adaptations thereof will occur to those skilled in the art, and all such variations, modifications and adaptations are considered to be within the scope of the invention, as defined herein, and by the claims appended hereto and equivalents thereof. What is claimed is: